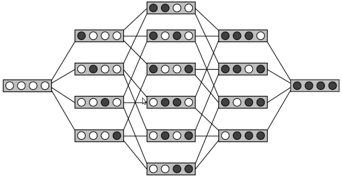
Supervised Learning

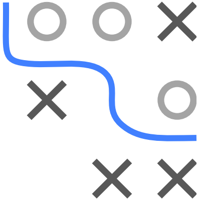
Wrapper methods



Learning goals

- Understand how wrapper methods work
- Understand how they coudl help in feature selection
- Know their advantages and disadvantages

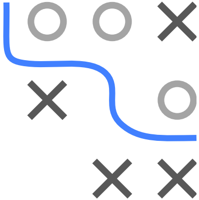
INTRODUCTION



- Wrapper methods emerged from the idea that different sets of features can be optimal for different classification learners.
- Given a set of features, we can use the classifier itself to assess their quality.
- We could just evaluate on the test set or use resampling techniques to achieve this.
- A wrapper is nothing else than a discrete search strategy for S, where the cross-validated test error of a learner as a function of S is now the objective criterion.

INTRODUCTION

There are a lot of varieties of wrappers. To begin with we have to determine the following components:



- A set of starting values
- Operators to create new points out of the given ones
- A termination criterion

INTRODUCTION

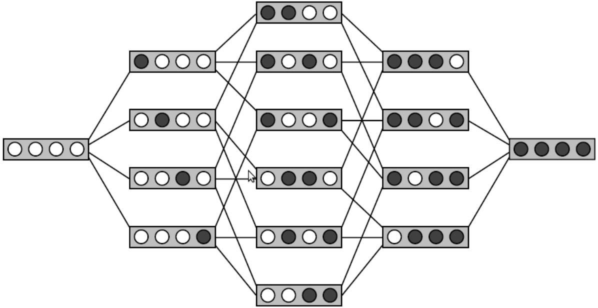
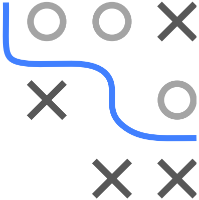


Figure: Space of all feature sets for 4 features. The indicated relationships between the sets insinuate a greedy search strategy which either adds or removes a feature.

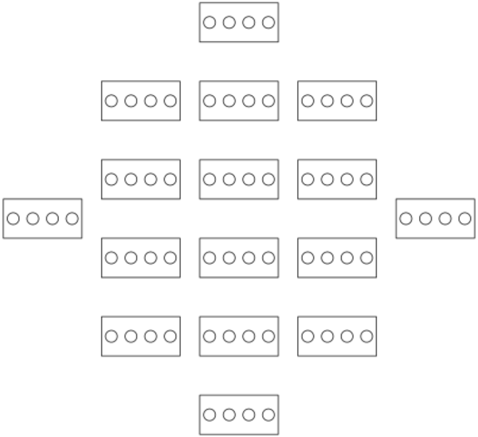
GREEDY FORWARD SEARCH



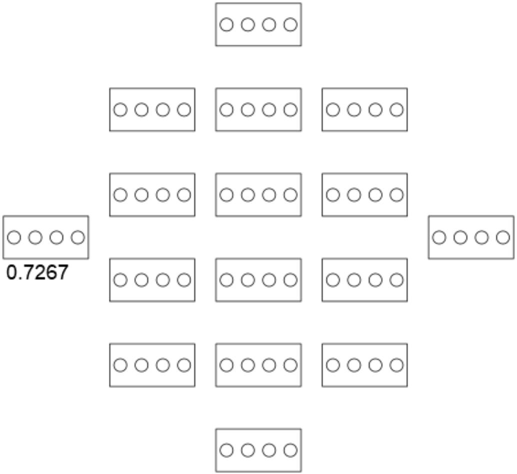
- Let $S \subset \{1, \dots, p\}$, where $\{1, \dots p\}$ is an index set of all features.
- Start with the empty feature set $S = \emptyset$.
- For a given set S, generate all $S_j = S \cup \{j\}$ with $j \notin S$.
- Evaluate the classifier on all S_j and use the best S_j.
- Iterate over this procedure.
- Terminate if:
  - the performance measure no longer shows relevant improvement,
  - a maximum number of features is used, or
  - a given performance value is reached.

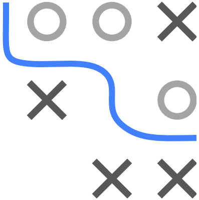
GREEDY FORWARD SEARCH

Example for greedy forward search on iris data:

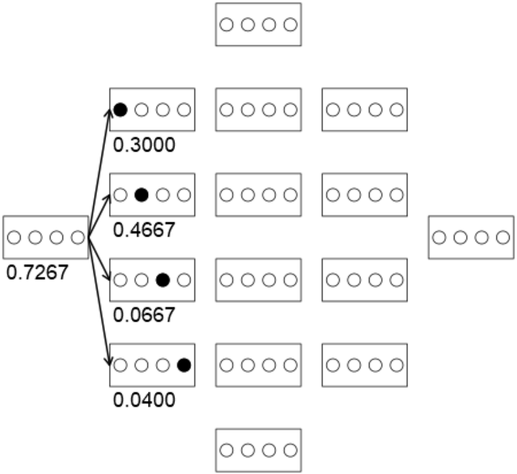


GREEDY FORWARD SEARCH

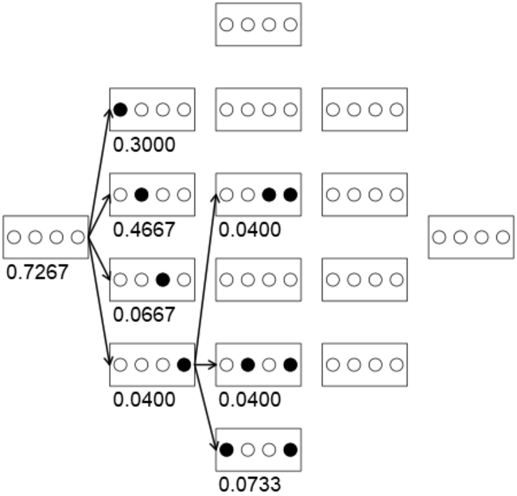


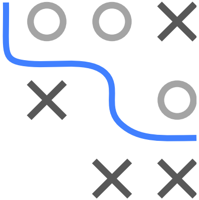


GREEDY FORWARD SEARCH

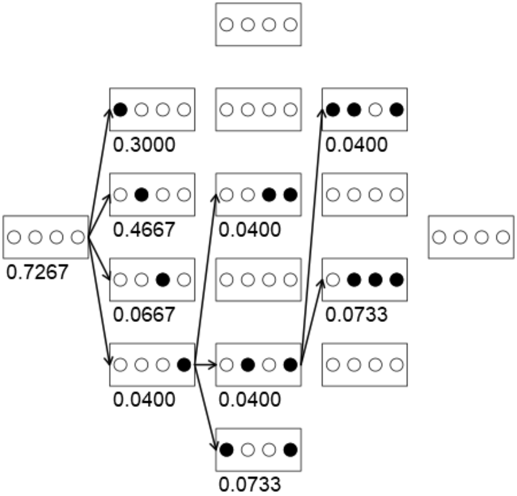


GREEDY FORWARD SEARCH

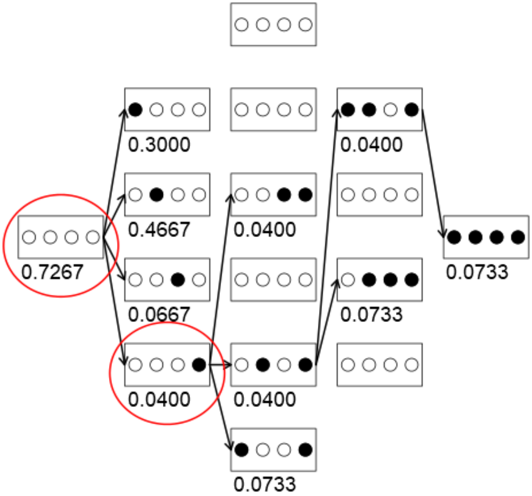




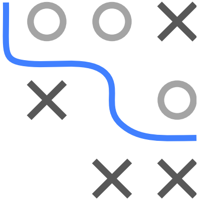
GREEDY FORWARD SEARCH



GREEDY FORWARD SEARCH

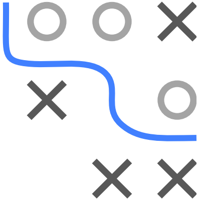


GREEDY BACKWARD SEARCH



- Start with the full index set of features $S = \{1, \ldots, p\}$.
- For a given set S generate all $S_j = S \setminus \{j\}$ with $j \in S$.
- Evaluate the classifier on all S_j and use the best S_j.
- Iterate over this procedure.
- Terminate if:
  - the performance drops drastically, or
  - a given performance value is undershot.

EXTENSIONS

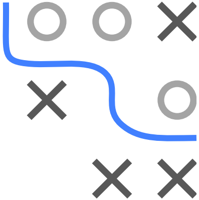


- Eliminate or add several features at once to increase speed.
- Allow alternating forward and backward search.
- Randomly create candidate feature sets in each iteration.
- Continue search based on the set of features where an improvement is present.
- Use improvements of earlier iterations.

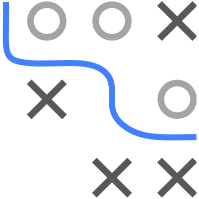
EXTENSIONS

Algorithm A simple 1+1 genetic algorithm

```
Start with a random set of features $S$ (bit vector $b$).
repeat
    Flip a couple of bits in $b$ with probability $p$.
    Generate set $S'$ and bit vector $b'$.
    Measure the classifier's performance on $S'$.
    If $S'$ performs better than $S$, update $S \leftarrow S'$, otherwise $S \leftarrow S$.
until One of the following conditions is met:
    • A given performance value is reached.
    • Budget is exhausted.
```



WRAPPERS



Advantages:

- Can be combined with every learner.
- Can be combined with every performance measure.
- Optimizes the desired criterion directly.

Disadvantages:

- Evaluating the target function is expensive.
- Does not scale well if number of features becomes large.
- Does not use much structure or available information from our model.